U.S.N.A. — Trident Scholar project report; no. 295 (2002)

Acquisition of 3-D Map Structures for Mobile Robots

by

Midshipman Edward H.L. Fong, Class of 2002
United States Naval Academy
Annapolis, Maryland

_____

(signature)

Certification of Adviser Approval

Assistant Professor Frederick L. Crabbe IV
Computer Science Department

_____

(signature)

_____

(date)

Acceptance for the Trident Scholar Committee

Professor Joyce E. Shade
Deputy Director of Research & Scholarship

_____

(signature)

_____

(date)

USNA-1531-2

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. REPORT DATE (DD-MM-YYYY) 07-05-2002	2. REPORT TYPE	3. DATES COVERED (FROM - TO) xx-xx-2002 to xx-xx-2002

4. TITLE AND SUBTITLE
Acquisition of 3-D Map Structures for Mobile Robots
Unclassified

5a. CONTRACT NUMBER
5b. GRANT NUMBER
5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Fong, Edward H. ;

5d. PROJECT NUMBER
5e. TASK NUMBER
5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME AND ADDRESS
US Naval Academy
Annapolis, MD21402

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME AND ADDRESS
,

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APUBLIC RELEASE
,

13. SUPPLEMENTARY NOTES

14. ABSTRACT
See report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:	17. LIMITATION OF ABSTRACT Public Release	18. NUMBER OF PAGES 66	19. NAME OF RESPONSIBLE PERSON email from USNA, Annapolis, MD, (blank) lfenster@dtic.mil
a. REPORT Unclassified b. ABSTRACT Unclassified c. THIS PAGE Unclassified			19b. TELEPHONE NUMBER International Area Code Area Code Telephone Number 703767-9007 DSN 427-9007

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39.18

REPORT DOCUMENTATION PAGE

Public reporting burden for this collection of information is estimated to average 1 hour per response, including g the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE 7 May 2002	3. REPORT TYPE AND DATE COVERED

4. TITLE AND SUBTITLE
Acquisition of 3-D map structures for mobile robots

5. FUNDING NUMBERS

6. AUTHOR(S)
Fong, Edward H. L. (Edward Hsiang Lung), 1980-

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
US Naval Academy
Annapolis, MD 21402

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
Trident Scholar project report no. 295 (2002)

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
This document has been approved for public release; its distribution is UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT: For an autonomous mobile robot to move around intelligently in its surroundings, it must possess a map of the environment which it can use to determine where it is, where it wants to go, and the best way to get there. The ability to successfully navigate in its surroundings allows the robot to perform more complicated tasks with greater autonomy. When a mobile robot is introduced into an unfamiliar environment, it does not have that map available and therefore must generate one itself. Methods currently exist to perform such a task in an environment where the robot moves about a single plane. However, this limitation restricts the robot's movement to a at and smooth surface (typically an indoor setting). This project builds upon current map building techniques to enable a ground-based mobile robot to navigate robustly in an unfamiliar, more three-dimensional (outdoor) environment. A new map structure has been developed to store three-dimensional information in a compressed form. It represents the robot's environment as a two-dimensional surface existing in three-dimensional space. This map structure was implemented on an all-terrain outdoor robot for use in urban environments. The map structure was tested both by comparing generated maps to the environment on which they were based, and by testing the robots ability to navigate with them. The map structure has been shown to accurately model the robot's environment and enable the robot to navigate in it while requiring few computational resources.

14. SUBJECT TERMS
mobile robots, 3-D map building, robot localization

15. NUMBER OF PAGES
66

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

Abstract

For an autonomous mobile robot to move around intelligently in its surroundings, it must possess a map of the environment which it can use to determine where it is, where it wants to go, and the best way to get there. The ability to successfully navigate in its surroundings allows the robot to perform more complicated tasks with greater autonomy. When a mobile robot is introduced into an unfamiliar environment, it does not have that map available and therefore must generate one itself. Methods currently exist to perform such a task in an environment where the robot moves about a single plane. However, this limitation restricts the robot's movement to a flat and smooth surface (typically an indoor setting). This project builds upon current map building techniques to enable a ground-based mobile robot to navigate robustly in an unfamiliar, more three-dimensional (outdoor) environment. A new map structure has been developed to store three-dimensional information in a compressed form. It represents the robot's environment as a two-dimensional surface existing in three-dimensional space. This map structure was implemented on an all-terrain outdoor robot for use in urban environments. The map structure was tested both by comparing generated maps to the environment on which they were based, and by testing the robots ability to navigate with them. The map structure has been shown to accurately model the robot's environment and enable the robot to navigate in it while requiring few computational resources.

Keywords: mobile robots, 3-D map building, robot localization

Acknowledgments

This project would not have been possible without the assistance and support of many wonderful people who spent countless hours helping me through my final year at the Academy. I would first like to thank God for the challenges He has created in this world to help us learn and grow. I would also like to thank my parents and siblings for all their help and support during the last twenty-one years of my life. I do not know where I would be today without them. Next I want to thank the research group at the Navy Center for Applied Research in Artificial Intelligence at the Naval Research Laboratory for providing me with the resources I needed to carry out my project - especially William Adams with whom my "Hey Bill, I got a quick question..." statements usually went on for hours. Last but most importantly, I would like to thank my adviser, Assistant Professor Frederick L. Crabbe IV for the guidance, support, patience, and longs hours he has spent helping me understand what was going on and pointing me in the right direction.

Thank you.

Contents

List of Figures

1 Introduction

1.1 Overview

The ability of a robot to move about autonomously in an environment can greatly increase the potential applications to which that robot can be applied. Whether it be moving boxes from one side of a room to another or exploring a distant planet, an autonomous mobile robot can be used for a greater range of tasks than its stationary counterparts. Many methods currently exist which provide a mobile robot with the ability to robustly navigate autonomously on a flat, smooth surface (essentially a two dimensional environment). However, such a restriction limits the environments in which a robot can operate. This project's focus is to extend the operable environments of a mobile robot by providing it with the ability to navigate autonomously in a more 3-D environment (i.e., one in which the ground is not perfectly flat). This project provides a robot with the ability to determine its location in 3-D space through dead reckoning and localization techniques, and the ability to represent its environment in a 3-D map structure for use in navigation tasks.

1.2 Teleoperational Control

In general, robots were created to take over tasks that humans consider to be dull, dangerous, or dirty [9]. Unfortunately, the current state of technology does not come close to providing most robots with the reasoning and perception capabilities that they

would require to robustly carry out their assigned tasks. As a result, most robots are teleoperated—where a human is controlling the actions of a robot remotely, usually from an area where he cannot directly see the robot. As the direct controller of the robot, the operator can incorporate many human qualities such as ingenuity, perception, and decision making into the actions of the robot. For example, if an autonomous robot observes a dark region in its environment, it would have to discern what it is. The difference between the robot identifying it as a shadow or a deep hole could determine whether it moves onto solid ground or plummets to its destruction. For a human, identifying such an area would be relatively easy. The operator could look around and determine whether anything around it could be casting that as a shadow. The operator could also have the robot pick up a rock and throw it at the black region, observe what happens, and draw a conclusion from that. The ability for a human to easily perceive its surroundings and determine a course of action makes teleoperational control an appealing solution for controlling robots.

1.3 Teleoperational Disadvantages

Unfortunately, there are many inherent disadvantages when a robot is teleoperationally controlled. With teleoperation, high communications bandwidth is required to send information between the human operator and the robot. The human needs to know the robot's current situation and status to make a decision as to its next course of action. The more information that the operator receives, the better the decision he can make. This requires a high bandwidth. Additionally, the operator needs to send commands to

the robot to tell it what to do. Communications delays can negatively impact the robot when the distance between the operator and the robot is vast. For example, if a robot on Mars is moving towards a cliff, the human operator on Earth would not know that fact for at least sixteen minutes after the cliff was first detected by the robot's sensors. If the operator then sends a command for the robot to stop moving, the robot would not receive it for at least another sixteen minutes. Within this thirty-two minute delay, the robot could have fallen off the cliff before it was stopped. An additional problem with teleoperation occur when some robots are so complicated that they might require multiple operators to control it, thus drawing on human resources. Finally, operators must sit and stare at a monitor to control the robot—a task that can become fatiguing after a long time. This defeats one of the purposes of creating robots in the first place—to relieve humans of tasks that we consider to be dull.

The solution to all of these problems would be to give the robot the ability to determine its own course of action—to make it autonomous. However, because of the current limitation in Artificial Intelligence (AI) technology, teleoperation is still considered the best method of control for tasks where [9, 13]:

1. The tasks are unstructured and not repetitive.

2. The task workspace cannot be engineered to permit the use of industrial manipulators.

3. Key portions of the task intermittently require dextrous manipulations, especially hand-eye coordination.

4. Key portions of the task require object recognition, situational awareness, or other advanced perception.

5. The needs of the display technology do not exceed the limitations of the communication link (bandwidth, time delays).

6. The availability of trained personnel is not an issue.

The development of the ability in robots to perform those and other tasks autonomously can prove beneficial. One of the most important abilities for a robot to possess is mobility. If a robot can be capable of moving about intelligently in an environment (especially an unfamiliar or dynamic one) on its own, it can then be given more complex and productive tasks such as searching and gathering information in dangerous and hostile areas.

1.4   Autonomous Mobile Robotics

The field of Autonomous Mobile Robotics is generally concerned with providing a mobile robot with the ability to move around intelligently in its environment. Four of the primary focus areas in this field are:

Exploration and Map Making — The ability to move into an unfamiliar environment and create a representation of that environment (to be used for Localization, Path planning, and Obstacle avoidance).

Localization — The ability to determine its position using a representation of the environment it is in.

Path planning — The ability to devise an efficient route between its current position and the desired goal.

Obstacle avoidance — The ability to avoid running into obstacles.

A robot that can perform these four tasks is able to determine (on its own) where it is, how to get to where it wants to be, and avoid running into any obstacles while moving towards

its destination. This would give the robot the capability of moving around without the assistance of a human controller. Therefore, the human (or perhaps another program) would only have to tell the robot where it needs to be, and then wait for the robot to arrive at its destination instead of controlling it every step of the way.

1.5 Project Goals

As will be discussed in Section 2, current methods of exploration, localization, path planning, and obstacle avoidance provide a mobile robot with the ability to navigate autonomously in an environment where it moves around on a single plane in space (equivalent to operating in a 2-D environment). This limitation restricts the robot's movement to a flat and smooth surface (typically an indoor environment). The goal of this project is to develop the capabilities for a ground-based mobile robot that will allow it to navigate in an unfamiliar, outdoor (more 3-D) environment. The robot must first be provided with the capability to determine its dead reckoning position in 3-D space. Next, it must be provided with a map structure which it can use to record pertinent information concerning its surroundings. This map structure must then be able to be used by the robot to determine where it is (localization), how to get to where it wants to go (path planning), and any hazards along the way (obstacle avoidance).

If a robot possesses the capability to maneuver itself in an environment where the elevation of the ground varies, then it can be used in a wider range of environments and tasked with more complex missions. For example, such a robot can be used to

assist outdoor military operations in urban terrain by moving ahead to search and gather information in dangerous and hostile areas. This project will extend the capabilities of the mobile robot developed at the Navy Center for Applied Research in Artificial Intelligence (NCARAI), located at the Naval Research Laboratory (NRL) in Washington, D.C. [14].

2   Background

2.1   Historical

The concept of creating a machine to carry out our biddings have been around since ancient times. For example, the Greek poet Homer once conceived of mechanical maidens built by Hephaistos (the Greek god of metalsmiths). In 1495, Leonardo da Vinci had created plans for a mechanical man [12]. However, it was not until the 1950s and 1960s (when transistor and integrated circuits were invented) that it became possible to even consider actually constructing a real robot. During those decades, most robots were either teleoperationally controlled or programed with a predetermined set of movements to perform (such as the robots used on a manufacturing line). It was not until 1967 that robots were created with the ability to determine their own movements based on their given goals. The following provides a brief overview of where artificially intelligent robots originated and some of the key advancements made since then.

2.1.1 Shakey

The first work on an artificially intelligent, autonomous mobile robot was performed at the Stanford Research Institute in 1967 [9]. In this work, a robot named Shakey, which employed a television camera, range finder, and an off-board computer to perform its calculations, was created. Shakey was able to perform some basic localization capability using a world model of its surroundings (which it was provided), its internal odometry, and the distance and angle to the boundary between the floor and the wall. The observed distance and angle would be compared to distance and angle that were predicted using its dead reckoned position and its map. The angular difference between the observed and predicted values was then used to determine and update Shakey's calculated position to its actual position. Due to limited processing speeds and sensors in the late sixties, it took Shakey over 10 seconds to localize itself. During this time, the robot did not move so that its positional corrections would be accurate. Shakey could also perform some basic path planning. It would choose to follow the path that consisted of the fewest number of turns (which was not necessarily the shortest path) to decrease the amount of computations it would have to perform. Due to Shakey's computational and hardware limitations, along with the methods it employed to perform localization and path planning, it was restricted to operate in only certain environments (smooth surfaces with a visible intersection between the floor and the wall) and was unable to find its goal in complex ones. However, given the time it was created, Shakey provided a tremendous advancement in the field of robotics [3].

2.1.2   Work by Hans Moravec

In the late 1970s, Hans Moravec (known as the father of AI robotics) was the first to successfully incorporate the use of stereo vision for navigation—employing it on The Stanford Cart. Though the Cart only moved about one meter every 10-15 minutes (with run times taking up to five hours), it was able to successfully detect, record, and plan paths around obstacles through the use of stereo vision [1].

In the early 1980s, Hans Moravec incorporated the use of evidence grids on a robot named Neptune. The information gathered about the robot's surroundings, through the use of its stereo vision and sonars, was stored in these grids which provided the robot with a map of the environment it is in. Incorporation of the evidence grids, which are discussed in the next section, provides the robot with the capability to generate its own map of the environment it is in—which it can then use to perform localization, path planning, and obstacle avoidance. Currently, most robotics research utilizes an evidence grid to help the robot navigate in its surroundings.

2.2   Current Progress

Mobile robotics has come a long way since 1967. There are currently many different robust techniques being employed by researchers to take care of their exploration, localization, path planning, and obstacle avoidance needs. In addition, technological advances in computational hardware and sensors have improved the performance and reliability of many robotic systems. However, the quest of creating an autonomous mobile robot is far
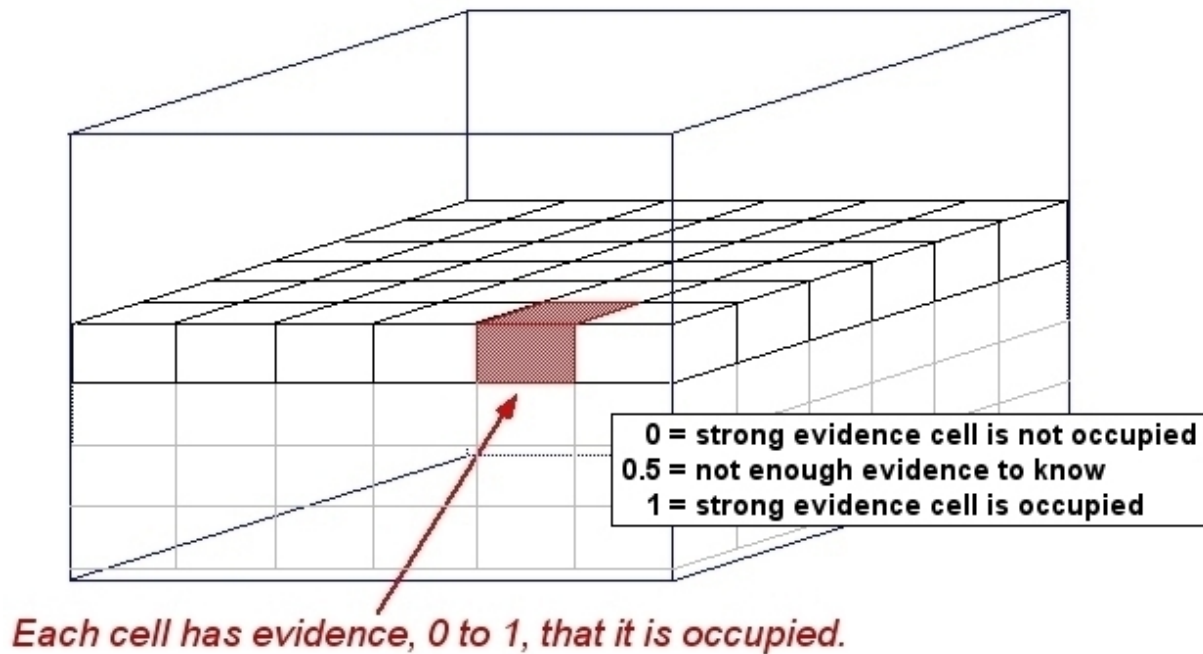
Figure 1: The Evidence Grid. (Figure courtesy of the Navy Center for Applied Research in Artificial Intelligence (NCARAI), Naval Research Laboratory (NRL))

from over. The algorithms currently in use are not perfect and could certainly be built upon to make a robot more robust and capable. The methods described here are the ones currently implemented on the robots at NCARAI—they will provide the basis for our extensions into three-dimensional movement.

2.2.1 The Map Structure

As discussed above, the most common structure used to represent the robot's environment is the evidence grid, as shown in Figure 1 [7]. The space surrounding a robot is divided into a number of cells arranged in a grid. Associated with each cell is a value between 0 and 1. The value in each cell is used to represent probabilities that a cell is occupied or unoccupied. Since an area in space can only be occupied or unoccupied at

any given time, one probability value can be used to represent both. If p represents the probability that a space is occupied, then $1 - p$ would represent the probability that the same space is unoccupied. A probability value of 1 indicates that there is a strong chance that the space is occupied. If a cell contains a probability of 0, there is a strong chance that the space is unoccupied (or very little chance that the space is occupied). The midpoint between these two values (0.5) is used to represent an area where there is an equal chance of the space being occupied or unoccupied—in other words, the occupancy of the corresponding location in the world is not known.

When the robot is placed in an unknown environment, all the cells in the grid are initialized with a value of 0.5. The robot uses the value of 0.5 to represent areas where it has not been to yet. This will later help the robot to determine where it should move to explore (map out) an unfamiliar environment. This map representation structure provides the robot with enough information to plan possible paths and move around—the robot only needs to know what areas it can move to and where it cannot.

2.2.2 Sensor Noise

The reason why the evidence grids represent the world as a series of probabilities is mainly to account for *Sensor Noise*. The term Sensor Noise is used to express the inaccuracy that exists when a sensor is scanning the world around it. Though the sensors used today are far more accurate and reliable than the ones back in 1967, they are still not (and might never be) perfect. Placing a sonar, or laser measurement system, at a fixed

distance away from an obstacle will provide multiple different distance readings over time. The situation is made worse if the angle at which the sonar ping (or laser beam) hits the surface of the obstacle is shallow enough that it is reflected away from the sensor (instead of back at it). In this case, the ping (or beam) might bounce off a secondary object and return to the sensor a moment later. Since these sensors determine distance by the time of flight of the ping (or beam), the delay makes the sensor believe that the obstacle is further away than it actually is. In addition, sonars are only able to detect obstacles within its field of view (which starts at the sonar and extends outward in a cone) and determine a distance to it. They do not have the capability to determine exactly where in its field of view the obstacle actually is—causing it to increase all the occupancy values in the grid which are within its field of view at the distance obtained. To complicate matters even more, the density of the air (since that affects the sonar wave's time of flight) and the reflectiveness of the object can also have an effect on the accuracy of the sensor readings.

The examples provided are just some of the problems affecting the reliability of sensor readings. It is therefore necessary to be able to use the information obtained from the sensors but also account for the fact that the values might not be 100% accurate. This is accomplished through the use of probabilities. There are currently many methods available to update the grid probability values with sensor readings. Some of these techniques include the Dempster-Shafer Theory [11] or *Histogrammic in Motion Mapping* (HIMM) [2]. The most commonly used method for updating probability values in evidence

grids is by using a recursive version of *Bayes' rule* [8, 9]:

$$P(H|s_n) = \frac{P(s_n|H) \times P(H|s_{n-1})}{(P(s_n|H) \times P(H|s_{n-1})) + (P(s_n|\neg H) \times P(\neg H|s_{n-1}))} \qquad (1)$$

$P(H|s_n)$ is the probability that there is an object in a cell, given that the sensor returned s_n. Likewise, $P(s_n|H)$ is the probability that the sensor would return the reading s_n given that the cell is occupied (H). $P(s_n|H)$ is the sensor model that was built beforehand by testing the sensor to determine all possible values of H. $P(H|s_{n-1})$ represents the probability that the cell is occupied from all previous sensor readings (the value currently stored in the grid). The $\neg H$ represents the probability that the space is not occupied. This can easily be computed considering that $\neg H = 1 - H$, making $P(s_n|\neg H) = 1 - P(s_n|H)$ and $P(\neg H|s_{n-1}) = 1 - P(H|s_{n-1})$. This is the updating method used by the NCARAI group at NRL for their experiments.

2.2.3   Exploration and Localization

If a robot were to be placed in an unfamiliar environment, for which it does not have a record, it would have to create a map in order to move around intelligently in that environment [14]. To create this map, the robot must perform sensor sweeps at various locations to determine where any obstacles are. The predicament is that in order to generate a map, the robot must be able to determine where it is (so it can incorporate its multiple sensor readings into the correct position in the map), in order to do that, it needs a map. The issues of exploration and localization are closely tied to each another to
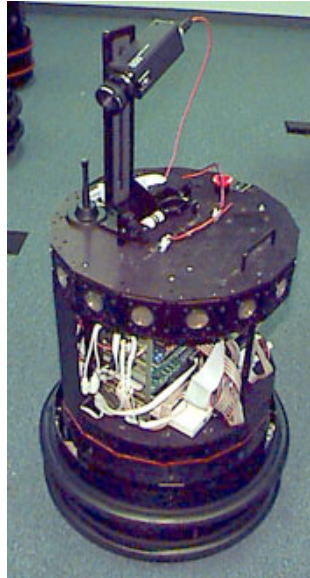
Figure 2: Nomad 200 Robot. (Photograph courtesy of NCARAI, NRL)

produce a working map of the environment that the robot can then use to move around intelligently.

The team at NCARAI has developed an efficient algorithm to resolve this dilemma. The group uses a Nomad 200 robot, shown in Figure 2, which implements the mobile robot system ARIEL (Autonomous Robot for Integrated Exploration and Localization) for their experiments. The robot's sensor suite consists of two bump detectors, 16 infrared sensors, 16 sonars, and a structured light range finder. The ARIEL system requires the integration of the robot's exploration and localization capabilities to map its surroundings and move around at the same time. When placed in an unfamiliar environment, the robot generates a sonar image of its immediate surroundings (the area within the range of its sensors). Using that image, it is able to determine where its frontiers are (i.e. the border between its recorded and unrecorded regions)—(Figure 3). It can then move to those frontier
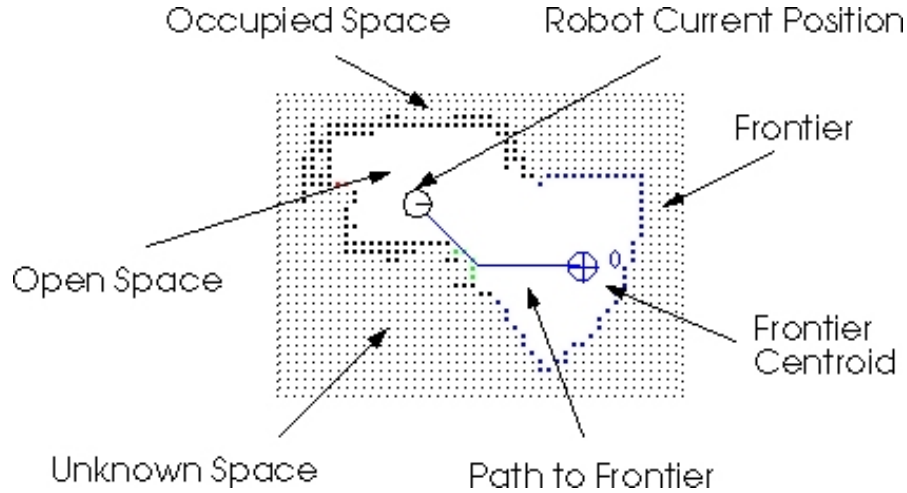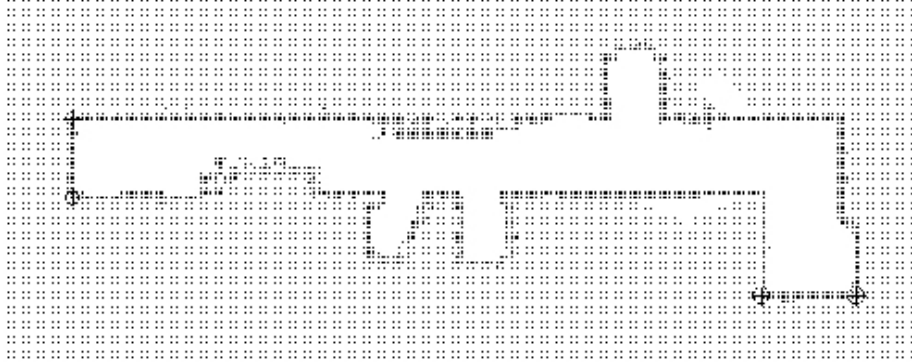
Figure 3: Frontier Exploration. (Figure courtesy of NCARAI, NRL)

locations and rescan the environment. This technique is known as *frontier exploration.* The robot continues to identify and move to new frontiers until all the areas within its traversable boundaries have been scanned, thus creating an initial map.

One of the most basic ways in which a mobile robot estimates its position is through dead reckoning. Using this method, the robot approximates its orientation and position with the information it receives from its motors. The robot can determine how much it turned, how far it traveled, and what direction it is facing based on the direction, speed, and duration each motor was turning. However, as the robot transits from one point to another, differences occur between its calculated position (through dead reckoning) and its actual location. Such disparities occur as a result of the robot's imperfect motor controls and its interaction with the real world (where the robot's position could shift due to environmental factors such as a slippery surface). The longer the robot maneuvers based solely on its plotted dead reckoning position, the less accurate its estimated

position becomes. Adding information to a map requires that the robot know its position within that map. However, if the robot calculates its position based solely on its internal odometer, it is likely that the map it generates will be inaccurate and unusable. The diagram displayed by Figure 4(b) is a map of a hallway that was generated through dead reckoning and exploration. When compared to what the map is suppose to look like, in Figure 4(a), we see the need for the robot to be able to determine its position with greater accuracy.

To maintain an accurate estimation of its position, the robot employs a method known as *continuous localization* [14]. This process requires that the robot generate a series of short-term maps of its surroundings. The robot first creates a new short-term map of the local area and updates it using its current sensor readings. After a predetermined amount of movement (or time), it creates another short-term map that it updates with its new sensor readings. It also updates all the short-term maps created before it. As the robot creates more and more short-term maps, it will eventually reach the maximum number of maps allowed (set ahead of time). Before the robot creates another map, it takes the most "mature" map (the oldest one) and uses it to determine where the robot actually was at that time by comparing its short-term map to its long-term map by means of a registration process. It then takes the x, y, and yaw (turn) offset (determined by the registration process) to correct its odometry. The short-term map used for this localization is then discarded and a new map is created. The *continuous localization* process is illustrated in Figure 5. The reason why short-term maps are continuously created, used for a brief

(a) Actual map of hallway.
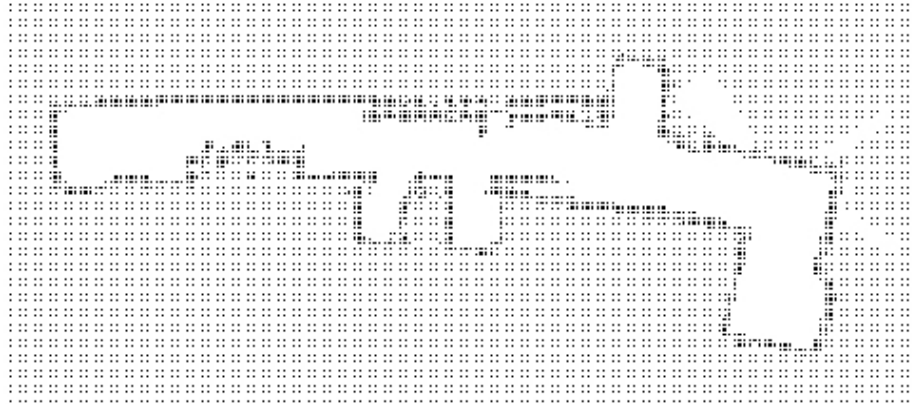
(b) Frontier-based exploration without localization.
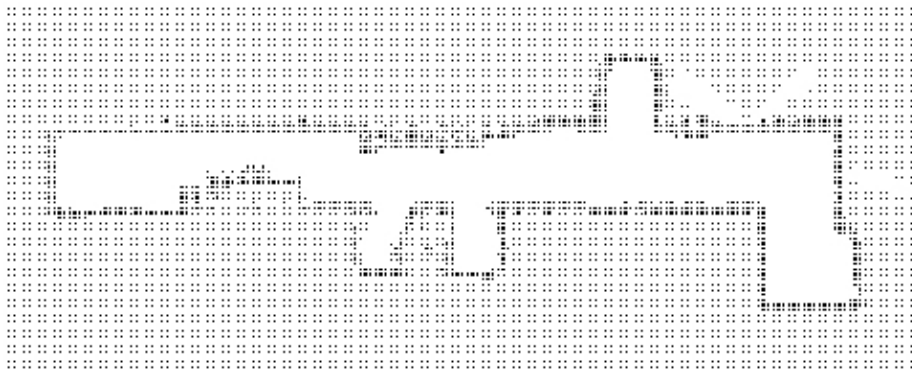
(c) Frontier-based exploration with localization.

Figure 4: Exploration and Localization: 3 Maps of a Hallway. (Figures courtesy of NCARAI, NRL)
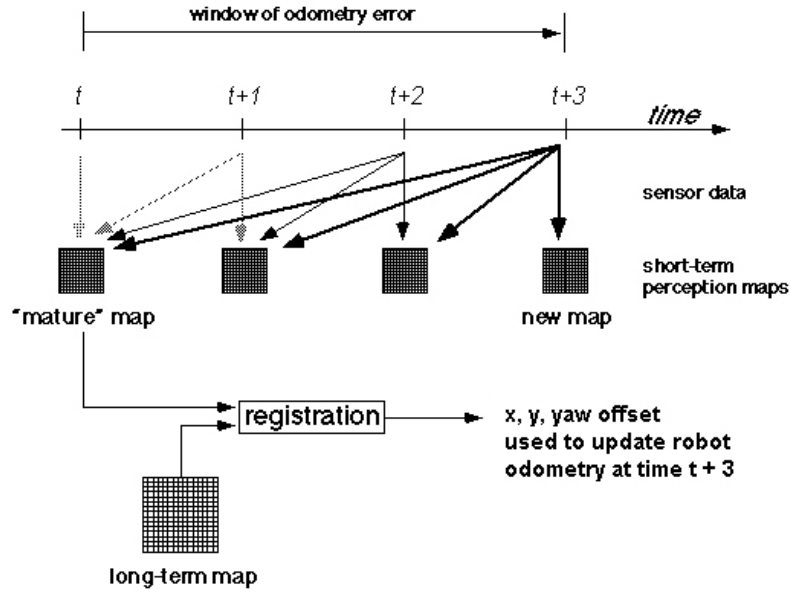
Figure 5: Continuous Localization. (Figure courtesy of NCARAI, NRL)

duration, and then discarded is because the robot's internal odometer is fairly accurate when determining its position for short distances but becomes more and more unreliable as its movements increases. If a robot performs a scan of an area without moving, its profile of the world around it will be very accurate (since it has incurred no odometric error). However, the map it creates (from its sensor readings) will be nothing more than a few scattered points). Therefore, the robot needs to move around to create a more detailed (and usable) map. As the robot initially moves around, it will incur a small odometric error while updating its map. If this error is small enough, the map produced will still be fairly accurate. Eventually, these small errors will accrue until they can no longer be ignored. Any additional sensor readings added to the map at this time would only distort it. It is at this point that the robot must take its short-term map (with the

level of detail it was afforded to gather by this time) and update its position by comparing it with the long-term map through a registration process. By constantly creating a series of these fairly accurate short-term maps, the robot is able to accurately estimate (and correct) its position while moving around.

The robot also has the ability to change its long-term map based on its short-term images [14]. Once the robot determines the positional offset between its mature and long term map, it can then combine the two to create an updated long-term map. Since the short-term maps are the most recent representations of the environment, it might contain some information which was not available when the long-term map was created. By being able to update its long-term map, the robot is more capable of adapting to a changing environment. For example, if a certain area along a robot's path is suddenly blocked, it will be able to record that information in its short-term map. Meanwhile, the robot's long-term map still holds that area to be traversable. Once the two maps are combined, the long-term map will be updated with the blocked passage and that information can be used in the future.

With this ability, the robot is able to fine-tune its main map through multiple sweeps from different positions. With each sweep and alteration, the long-term map becomes more accurate, enabling the robot to perform a more precise estimation of its location. Figure 4(c) shows the map of the same hallway generated though the use of exploration and localization.

Unfortunately, the current continuous localization algorithms are not too robust. If the

registration process produces an inaccurate correction, the robot will be moved sightly off track. Subsequent registrations might be able to correct for the previous error. However, if the error is great enough, then the following registrations would also fail—causing the robot to lose an accurate estimate of its position. If the robot is also updating its long-term map with the short-term ones created, then the long-term map will become more and more distorted and end up unusable.

2.2.4   Path Planning

The path planning method employed by the robots at NCARAI is a wavefront propagation based algorithm called Trulla [10]. This algorithm starts with the location of the goal. The algorithm then examines all the unoccupied immediate neighbors of the goal and sets them to "point" to that cell, as show in Figure 6. The algorithm then makes the next set of neighbors point to the previous set. The algorithm will continue to "radiate" outward from the goal until all accessible, unoccupied cells are marked with a direction along a path that will lead to the goal. This will create a map so that the robot, no matter where it is placed, will be able to find its way to the goal by following the field lines.

One benefit of computing the path for each location on the map can be seen when a robot has to divert from its path to avoid an obstacle that was not on the map. After reactively steering clear of that obstacle, the robot can then pick up the path from its new location. Another advantage of this algorithm is that undesirable areas can be weighted
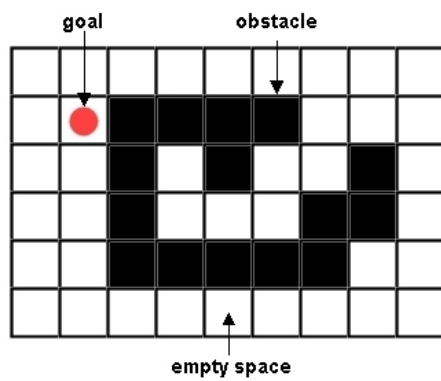
(a) Map with Goal.
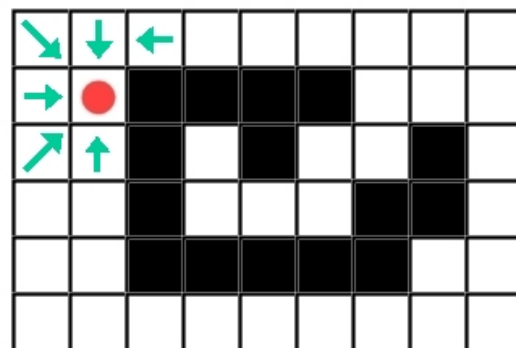
(b) Immediate Neighbors.
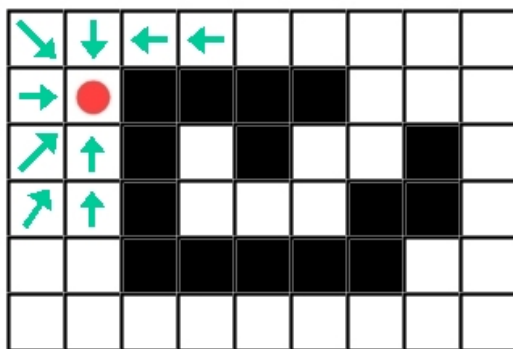
(c) Next Set of Neighbors.
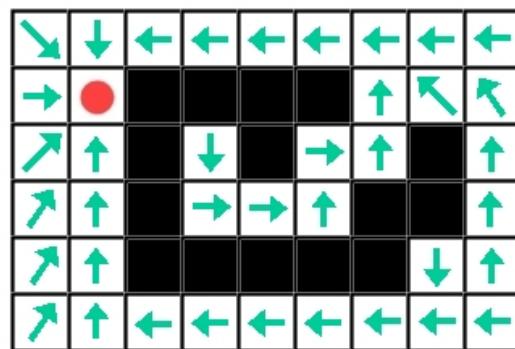
(d) Map with All Paths Drawn.

Figure 6: TRULLA.

to slow the wavefront propagation through that area. This would allow paths which are more desirable to spread faster and reach more cells than undesirable paths. Once a cell has been marked, it will not be changed (since there might be more than one path leading to the goal from a single location). Using this method, the more desirable path would reach and mark that cell first. However, the most desirable path could be a shorter path through an undesirable region. Fortunately, the wavefront propagation "naturally handles the trade-offs between a longer path through desirable terrain versus taking shortcuts through less desirable terrain." [10]

2.3   Current Limitations

All of the development and testing of the mobile robots at NCARAI are currently being performed within a controlled laboratory where the robots move around on a smooth, flat surface. By taking advantage of an environment in which the robots can perceive in just two dimensions, the task of developing their exploration and localization capabilities were simplified. By eliminating many variables, the Nomad 200 robot could focus its limited computational power (an Intel 120 MHz Pentium processor) on a more tractable problem [3]. In this environment, the smooth, flat floor of uniform composition can be represented as a single plane. The obstacles and walls have angular contours and are made of reasonably reflective materials. However, such an ideal environment is rarely found outside a laboratory environment where a mobile robot will likely be used.

A primary concern with introducing a robot into a three-dimensional environment,

such as an outside environment, is that the robot will be able to traverse on more than one plane. With this added complexity, the robot will also need to have the capability to determine its roll, pitch, and elevation. From this data, it can determine its three-dimensional location and orientation in the environment and update its world model (its map). Once the robot is recording three-dimensional information, the map representation structure must be changed to be capable of storing such data. Its localization algorithm will also have to be modified to work with the new map structure.

2.4   Related Work

There has been some research performed in the area of creating a 3-D model of both indoor and outdoor environments. However, the 3-D environmental models of those projects are not the types of models that can easily be used to assist a robot in navigating in its environment. For example, a project [5] was designed to create a 3-D map of a corridor by moving the robot down the corridor while having it record a profile of the distance around it (in a plane perpendicular to its direction of travel). Using this data, the robot then merged the individual sensor readings to create polygons to represent those surfaces. The result was an accurate 3-D representation of the room from the ground up. The robot in that project was only able to move in a 2-D plane, which is not suitable for a robot moving in an outdoor environment. In addition, it took close to six hours to compute the 3-D map mentioned above. For a mobile robot which will need to generate its own map, having to wait a few hours before it can actually use its map would pose a serious

problem.

2.5   The Required Map Structure

The map representation structure that needs to be created must be able to store the 3-D information of the environment—this includes the presence of ramps, curbs, hills, ditches, and holes [9]. In addition, since the robot's altitude can change as it moves around, a situation could exist where the robot can be in the same x and y position in the world but at different altitudes. These conditions must be represented in such a way that the robot can easily extract the needed information from the map for it to perform its exploration, localization, path planning, and obstacle avoidance tasks quickly and efficiently. The 2-D map building algorithms currently implemented on the robots at NCARAI must be changed to receive 3-D sensor readings and incorporate that information into a map structure that will be able to represent 3-D information.

3   The Robot in a 3-D World

3.1   Robot Details

The robot being used for this project is the ATRV-Jr. (built by iRobot™)—as shown in Figure 7(a) [4]. The sensors on the ATRV-Jr. include 17 sonars, 2 SICK™ Laser Measurement System (LMS), a Dynamic Measurement Unit (DMU), and a TCM2™ Compass. Each of the 17 sonars return a value that provides 181 distance readings between itself and

(a) The Enhanced ATRV-Jr.
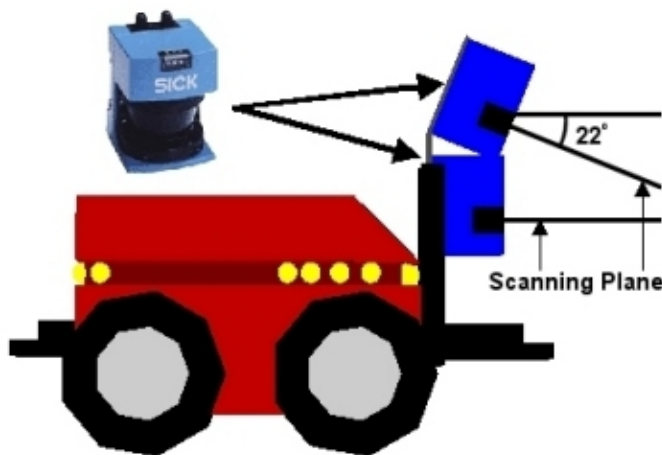
(b) Side View
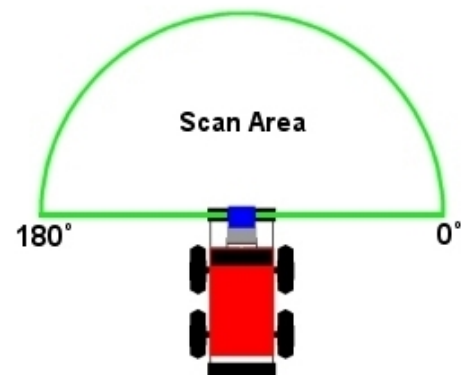
(c) Top View

Figure 7: The ATRV-Jr.

the nearest obstacle within the cone of its scanning region. These sensors came built-in with the robot but are not used in this project. The SICK LMS provides the distance between itself and the nearest obstacle within a scanning plane (as illustrated by Figure 7(b) and Figure 7(c)) from 0° to 180° with a 1° angular resolution—returning a profile of its scanning plane (similar to the image received from a radar). The bottom SICK is mounted parallel to the robot's base while the top one is mounted at a 22° downward angle, shown in Figure 7(b). The addition of the top SICK provides sensor readings on a different plane than the one on which the robot is moving. This enables the robot to sense changes in the ground elevation before it moves into that position—giving it a 3-D ground view of its environment. Without the additional SICK, the sensors will be limited to scanning a plane that is parallel to the robot. The DMU is an Inertial Navigation System (INS) which combines three rate gyros and three accelerometers to provide the robot with its acceleration (along the x, y, and z axes) and angular rotation rate (for its *roll*, *pitch*, and *yaw*). The TCM2 Compass provides a compass reading along with the robot's roll and pitch.

3.2 Roll and Pitch

While the robot is moving on just a flat, smooth surface (basically a single plane), it is able to determine its position and orientation by keeping track of (and updating) its x, y, and *yaw* (orientation) values. These three values are all that the robot needs to describe its *pose* (position and orientation in space). While moving around on a single

plane, it is able to ignore its altitude, roll, and pitch—thereby dramatically simplifying the computations it needs to perform in determining and updating its is position. However, now that the robot will not be moving on just a flat surface, the 2-D positional (x, y and yaw) update method is enhanced to account for the three remaining degrees of movement: $roll$, $pitch$, and z (altitude).

There are two means available on this robot for determining its $roll$ and $pitch$. The first is to take those values directly from the the TCM2. Though the values provided by this sensor are accurate, it takes approximately 3 seconds for the sensor to stabilize on the true reading. The other method is to use the DMU. To determine the $roll$ and $pitch$ from the DMU, the angular rotation rate for the x and y axes are used. With the angular rotation rates and the duration for which the robot is moving its motors, the change in $roll$ and $pitch$ can be determined and then incorporated into the previous values to obtain the current $roll$ and $pitch$. Unlike the TCM2, the DMU is able to provide its updated readings without any noticeable delay. The robot's local coordinate system used for these calculations is shown in Figure 8.

One drawback to using the DMU is its limited accuracy. Through multiple test runs, it was determined that the values obtained from the DMU are not accurate enough to be used for calculating the robot's $roll$ and $pitch$—regardless of the speed at which the robot is moving. For example, when the robot was sent up a ramp, it would at first correctly determine its angle of inclination. However, as the robot keeps moving up the ramp, the DMU would cause the inclination to drift (due to an actual non-constant drifting of the
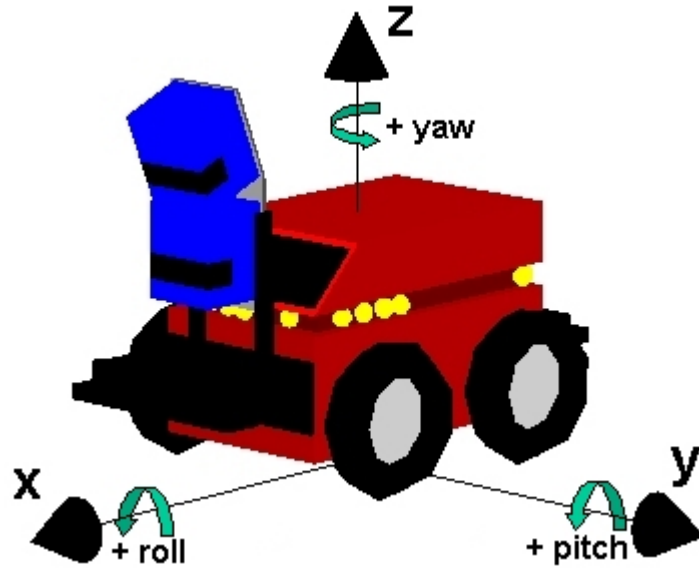
Figure 8: The Robot's Local Coordinate System.

DMU). Eventually, the robot would think it was going down the ramp when in-fact it was heading up. For this reason, the TCM2 is used instead while running the robot at slower speeds—to minimize the error incurred from the stabilization delay. The slower the robot moves, the less change there is to its inclination (since it could not have moved that far). As a result, its current *roll* and *pitch* would not be that much different to that of a few seconds ago—so while the TCM2 was stabilizing for past positions, it will be somewhere close to its current *roll* and *pitch* and could therefore be used (with some degree of error).

3.3 The 3-D Pose

With the ability to move and sense objects on multiple planes, the robot needs to know its position (x, y, and z) and orientation (*roll*, *pitch*, and *yaw*) in 3-D space. The robot's new *pose* description will now comprise of these six values. The values

for *roll* and *pitch* are already known—they are read from the TCM2 as described in Section 3.2. The value for *yaw* is obtained from the DMU using the same method described in Section 3.2 for determining the *roll* and *pitch*. The DMU is used to determine the *yaw* value because the ATRV-Jr.'s internal odometry determination for its own *yaw* value is extremely inaccurate, even more so than the DMU. For example, if the robot were to make a lap around a room, its internal odometry would place it approximately 40° off its actual orientation whereas the DMU would only be off about 5°. The compass is not used for the determination of the *yaw* since it can be easily affected by any magnetic disturbances in the area.

Once the orientation of the robot is determined, its position can then be calculated using some of the information from the robot's internal odometer. Even though its *yaw* value will be inaccurate, the distance traveled can still be used. The new x and y position of the robot cannot be taken directly from the odometer since it was determined by using the odometry's *yaw* value and does not account for the robot's *roll* or *pitch*. For example, if the robot traveled 2 meters in the x-direction while on a 30° upward pitch, the the actual distance that it moved along the x-axis would be: $2 \times \cos(30°) = \sqrt{3}$ meters. In addition, it would also have gained a height of: $2 \times \sin(30°) = 1$ meter. The z position must be calculated using the robot's *roll* and *pitch* since there is no altimeter available that would be capable of providing the robot with its true altitude. Therefore, estimation of the robot's position (x, y, and z) must use the internal odometer as well as the information concerning its *roll*, *pitch*, and *yaw*.

The internal odometer provides the robot with its estimated x and y position in its local frame of reference (not the global map frame)—which we can convert into the distance traveled:

$$distance = \sqrt{(\Delta x)^2 + (\Delta y)^2} \tag{2}$$

From this information, the orientation, and robot's last position, its new pose can be calculated using the following basic transformation matrices [6]:

To move the robot a, b, and c, units in the x, y, and z direction respectively:

$$Trans(a, b, c) = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

To rotate the robot ψ degrees about its x-axes $(roll)$:

$$Rot(x, \psi) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\psi) & -\sin(\psi) & 0 \\ 0 & \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

To rotate the robot θ degrees about its y-axes $(pitch)$:

$$Rot(y, \theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

To rotate the robot ϕ degrees about its z-axes (yaw):

$$Rot(z, \phi) = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 & 0 \\ \sin(\phi) & \cos(\phi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

Using Equations 3-6, a transformation matrix (T) describing the robot's *pose* in space can be determined using the following equation (where (a,b,c) represents the robot's location

(x,y,z) in space before it started moving, and ψ, θ, and ϕ represents the current *roll*,

pitch, and *yaw* respectively):

$$T = Trans(a, b, c) \times Rot(z, \phi) \times Rot(y, \theta) \times Rot(x, \psi) \tag{7}$$

Since the matrix T describes where the robot was and which way it is pointing, moving

it d distance (calculated from Equation 2) along the its x axis (the direction which it is

pointing) will result in a transformation matrix describing the robot's new pose:

$$T \times Trans(d, 0, 0) = \begin{bmatrix} o_1 & o_2 & o_3 & X_n \\ o_4 & o_5 & o_6 & Y_n \\ o_7 & o_8 & o_9 & Z_n \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

where X_n, Y_n, and Z_n is the robot's new 3-D position (and o_1-o_9 is the orientation in

matrix form).

This is the method used to determine the robot's 3-D *pose* for this project. Its position $(x, y, \text{and } z)$ is determined from Equation 8 and its rotations are the ones used in Equation 7. A "POSE" module was created which uses the method described above to serve as the new 3-D internal odometer for the robot. It obtains the required data from the DMU, TCM2, and the robot's 2-D (original) internal odometer. The updated *pose* $(x, y, z, roll, pitch, \text{and } yaw)$ is then determined and used by the robot as illustrated in Figure 9. All information needed concerning the robot's *pose* is now taken directly from this module—essentially making it the robot's new internal odometer.
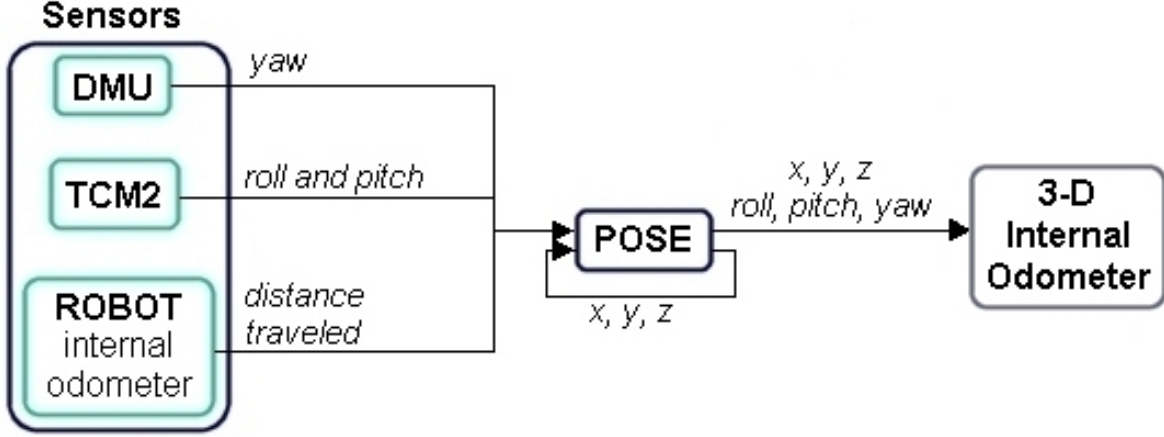
Figure 9: The POSE Module.

3.4   Determining the Location of Objects in 3-D Space

To be able to update information in the map from sensor readings, the world coordinates of the sensor reading must be determined in order to update the correct position in the map. This is done by extending the method that is used to determine the robot's position in 3-D space. Using the 3-D *pose* of the robot (obtained from the "POSE" module) and the location of the sensor on the robot (in relation to the robot's local coordinate system (LCS)—Figure 8), the scanning origin (x, y, and z position of the sensor in world coordinates) and orientation (*roll*, *pitch*, and *yaw* of the sensor) can be determined. Using (x_r, y_r, z_r, ψ_r, θ_r, ϕ_r) to represent the robot's *pose* (x, y, z, *roll*, *pitch*, and *yaw* respectively) in relation to the world coordinate system, the transformation matrix for the robot (T_r) is as follows:

$$T_r = Trans(x_r, y_r, z_r) \times Rot(z, \phi_r) \times Rot(y, \theta_r) \times Rot(x, \psi_r) \tag{9}$$

Computing the transformation matrix for the sensor (T_s) in terms of the robot's LCS would be (using x_s, y_s, z_s, ψ_s, θ_s, ϕ_s to represent the sensor's position and orientation in

the robot's LCS):

$$T_s = Trans(x_s, y_s, z_s) \times Rot(z, \phi_s) \times Rot(y, \theta_s) \times Rot(x, \psi_s) \qquad (10)$$

By multiplying these two transformation matrices, the *pose* of the sensor in the world coordinate system $(T_{s'})$ can be determined: $T_{s'} = T_r \times T_s$. The readings obtained from the sensors are provided in the LCS of the sensor. For example, the SICK returns 181 (x, y) values for each scan from 0° to 180°. These values represent the x and y distance of the sensed obstacle from the sensor's origin. Since $T_{s'}$ represent the sensor's actual world coordinates (and orientation), converting the sensor readings from the sensor's LCS to its world coordinates would require one final transformation. Using x_o and y_o to represent the x and y distance of the object from the sensor, T_d (the transformation matrix from the sensor to the object) would be: $T_d = Trans(x_d, y_d, 0)$. Multiplying $T_{s'}$ by T_d would then give you T_o (the position of the obstacle in the world coordinate system).

Using this method, the robot is capable of determining the 3-D location of any object it picks up from its sensors—enabling it to add the object to its 3-D map.

4   Map Representations

This section will first briefly review the 2-D evidence grid currently used for localization and navigation. An expansion of the evidence grid into a 3-D version will then be presented, followed by an explanation of why such a map structure will not provide a feasible solution for robot navigation in a 3-D world. Finally, the creation of a $2\frac{1}{2}$-D map structure is provided and evaluated. It will show that the $2\frac{1}{2}$-D structure can ad-

dress many of the drawbacks of using a 3-D evidence grid and can provide many of the functionalities that a 2-D evidence grid does for an indoor robot. It will also address the limitations present in using a $2\frac{1}{2}$-D map structure.

4.1   2-D Map Representation

One of the map representation structures used for mobile robot navigation is a one layered (2-D) evidence grid. As described in Section 2.2.1, this map structure is capable of accurately representing an environment where the robot only moves around on a flat, smooth surface. Figure 10 is a 2-D evidence grid representation of Room 105, Building 1, at NRL (the size of the room is approximately 13 meters x 13 meters and uses a 128×128 size evidence grid—16,384 cells). The black dots represent areas where the robot has not explored yet. The white areas represent unoccupied space while the blocks represent occupied space. This map structure works fairly well for the type of environment it represents, but for the reasons mentioned in Sections 2.3 and 2.5, the structure needs to be altered to accept 3-D information.

4.2   3-D Map Representation

One of the most direct ways to obtain a 3-D map representation structure is to use a 3-D evidence grid. Instead of using the cells in a 2-D array to store the occupancy probability for every (x, y) coordinate in space, the cells in a 3-D array can be used to record the same information for every (x, y, z) location in space. Figure 11(b) and 11(c)
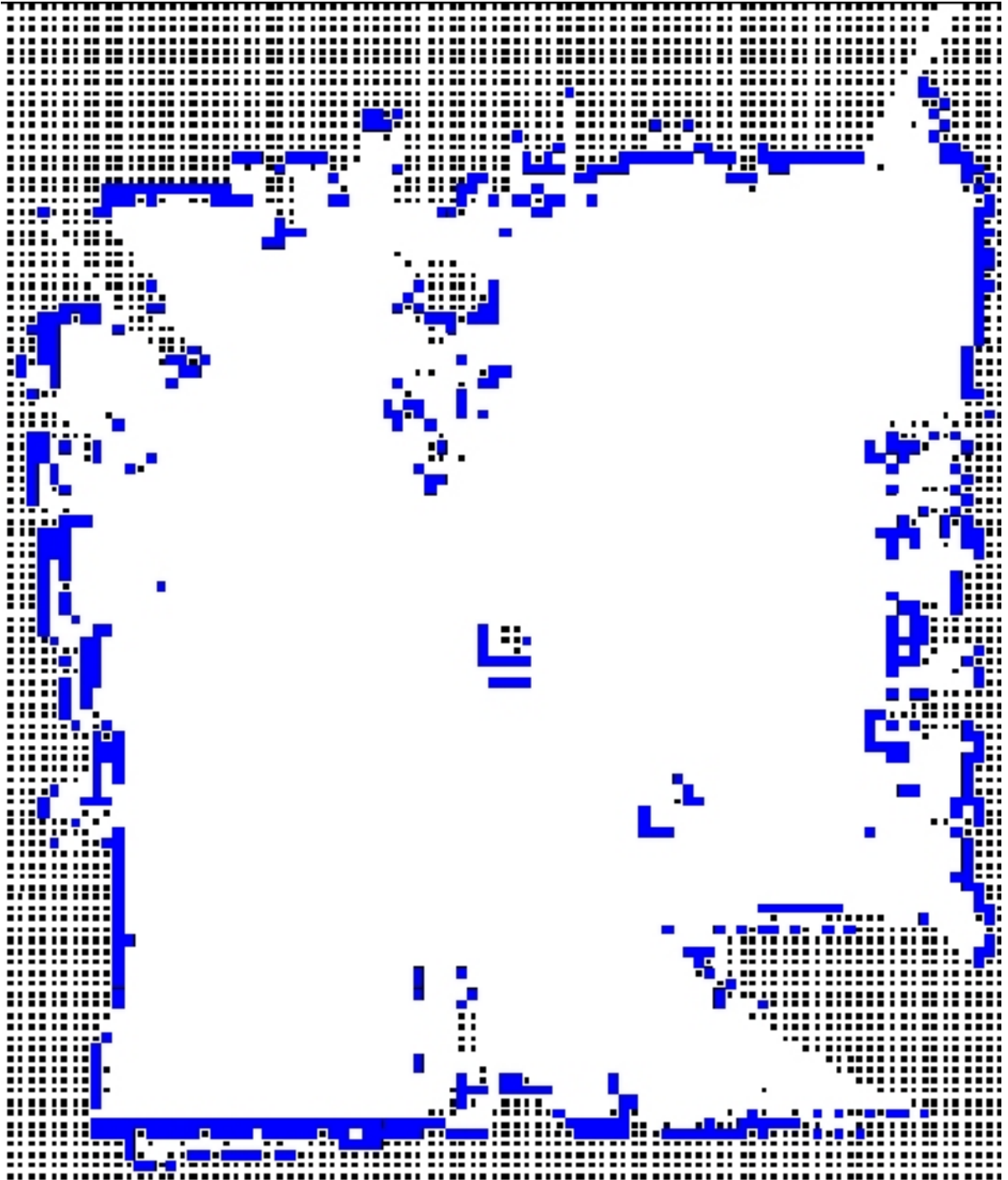
Figure 10: The 2-D Evidence Grid

are 3-D evidence grid representations of the same room introduced in Section 4.1. In these views, the dots representing "unknown" areas are not shown for a better view of the map. Although, this structure (as it currently stands) does store the 3-D information obtained from the robot's sensors, it uses a large amount of memory to store that map. The number of z layers needed to represent the environment is the factor by which the 2-D evidence grid is increased. This increases the amount of memory needed to store the map and requires more computations to access and use the map. In addition, the map in Figure 11(b) and 11(c) only displays a height of 1 meter with a 0.125 meter resolution. This map is stored in a $128 \times 128 \times 8$ evidence grid (131,072 cells) If a finer resolution is desired to better represent the variations in ground elevation, or if the robot moves above or below the map boundaries, the number of z layers would have to be increased—adding to the size of the map and the computations involved in accessing, retrieving, and using that map for navigation and localization.

Resolution along the z layer is especially important since it determines whether or not the robot can move from its current location to another area. A mobile robot such as the ATRV-Jr. (and many other wheeled robots that are restricted to moving about on a surface) has a limited ability to overcome vertical obstacles. It will not be able to roll over obstacles that are above a certain height. For example, the maximum climbing height of an obstacle that the ATRV-Jr. can roll over is about 0.1 meter. Using that information, any difference in elevation (from one cell to the next) greater than 0.1 meter would be considered an obstacle. Taking the robot's climbing height into account, the 0.125 meter

(a) Panoramic Photograph of Room.
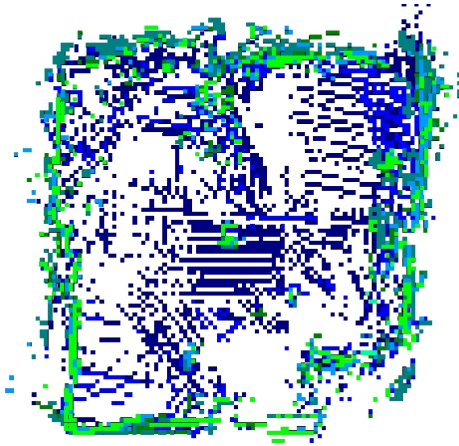
(b) Bird's Eye View.
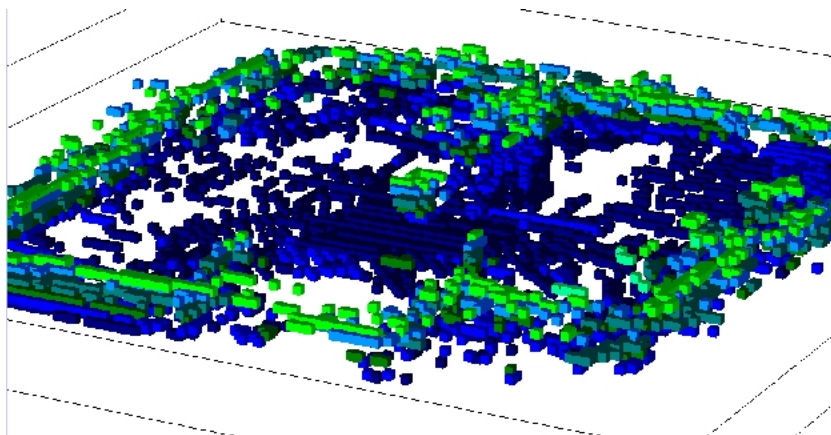
(c) View From Bottom-Left Corner of Figure 11(b)

Figure 11: Room 105 at NCARAI, NRL

(a) Height Change Not Detected.

(b) Change Mistaken to be Within Tolerance.

Figure 12: Hidden Obstacles.

cell resolution in the maps for Figures 11(b) and 11(c) would have to be made finer so that the robot can distinguish an obstacle. In Figure 12(a), an obstacle is hidden from the r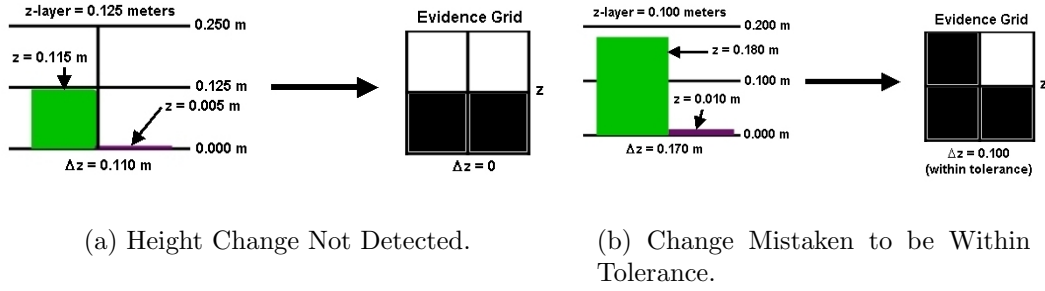obot as a result of the cell's resolution. Increasing the cell resolution to 0.1 meters will not work either, as shown in Figure 12(b), since the height used to represent each cell is taken from the center of the cell in world coordinates—the actual height of the object is not recorded (just the probability that there is an obstacle somewhere in that cell). In order for the 3-D evidence grid to work, each cell can represent (at the most) half of the maximum climbing height of the robot. For the ATRV-Jr, it would need a 3-D evidence grid with each z layer representing at most 0.05 meters. To represent the same area as in Figure 11(b) with the required resolution, a $128 \times 128 \times 20$ evidence grid (327,680 cells)—20 times the size of the 2-D evidence grid—is used. Even if such a grid was created, it would only provide the robot with information to navigate in a block of space 1 meter high. If the robot was to move above or below this 1 meter block, it would have to create another large evidence grid (or increase the size of its current one). The memory required to store such a grid, and the computational resources needed to access

and use such a large structure, makes it unfeasible to implement this map structure for 3-D movement.

4.3 $2\frac{1}{2}$-D Space

Another possible representation of the 3-D space is a two-layered, three-dimensional array. The first two dimensions are used to identify the x and y coordinates of the location in space—from a bird's eye view. The third dimension is broken up into two layers. The first layer is used to represent the relative height of the tallest obstacle at that location while the second layer stores the probability that the tallest object for that (x,y) location exists at that height. All heights are stored as a relative measurement in reference to the robot's initial position as ground zero. Using an integer to represent millimeters of height, a map can be created to provide the robot with information in the range of ±32 meters from the robot's starting altitude, with a potential accuracy of 1 millimeter.

All of the cells in both layers are initialized to a value of zero. For the first layer (height), this initialization does not really matter since its value is not used if the corresponding probability (in the second layer) is 0. If the probability value for any cell is 0 (or less than some predetermined value used to represent the minimum probability that must exist before the height in the first layer is considered to be valid) then the robot treats that cell as an unknown area and its z value is not used for any calculations. As the robot moves around, it is able to determine the location of objects around it (including the ground) and, using the method described in Section 3.4, find its x, y, and z world

coordinates. The robot would then update the corresponding cell of that x and y location in the grid with the z value and change the probability value (in the second layer) according to the sensor model—which is based on Equation 1. If an object is then detected at that (x,y) location with an altitude greater than the currently stored altitude, then the z and probability value will both be updated. The new recorded height will be the new z value while the updated probability would be based on both the previous height and probability, along with both the current height and sensor probability in the following way. If the difference between the two heights is within a certain set range, the new probability will be a combination of the previous map and current sensor probabilities. If the height difference is beyond the tolerance set, then the new probability will just be the probability of the sensor reading. This will allow the maximum height of an obstacle to be recorded and its probability increased if multiple readings are taken at or within a certain range of that height. Multiple detections of an object for that (x,y) location which are below the height range will not affect the probability value for that cell. Since the probability represents the chance of that location's height of being the value stored in the first layer, the exclusion of any readings below a certain predetermined range will prevent the probability value of that height from being prematurely increased by many low-level scans.

If a certain cell pair (height and probability) indicates a probability of a certain height, and a sensor reading indicates an empty space below that height (in situations where the first few readings were wrong or the object moved), then the probability for that location

will decrease according to the sensor model (the height will remain unchanged). If there is no object at that height, then the probability will keep decreasing until it reaches 0—at which point the height at that location is considered unknown. The next sensor reading which indicates an object at that (x,y) location will replace the previous value and update its probability according to the sensor model. Using this method, the map can be updated to account for a changing environment and erroneous sensor readings.

4.3.1   Obstacle Avoidance—The Ramp

The major drawback of using a 3-D evidence grid to represent an environment is the size of the grid required to represent changes in elevation (as described in Section 4.2). The ability to determine the elevation change from one cell to the next is crucial for the robot when determining if it can (or should) move from one area to the next. For a 3-D evidence grid, this will require the creation of a fairly large map (depending on the actual change in elevation experienced). However, for the $2\frac{1}{2}$-D map structure, this will only require twice the space of a 2-D evidence grid. To show that the $2\frac{1}{2}$-D map will actually represent changes in elevation that can be used by the robot, two maps were created of a ramp in back of Building 1 at NRL (as pictured in Figure 13(a)). The first map (Figure 13(b)) is a 3-D evidence grid using a z cell size of 0.125. As described in Section 4.2, this map will not allow the robot to accurately differentiate some ramps from obstacles or even identify many obstacles. The fact that each cell represents 0.125 meter (higher than its 0.1 meter of climbing ability) will prevent the robot from moving
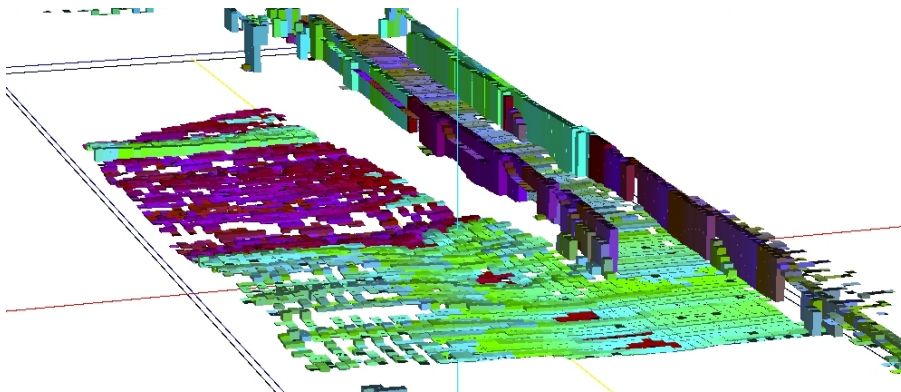
(a) Photograph of ramp.
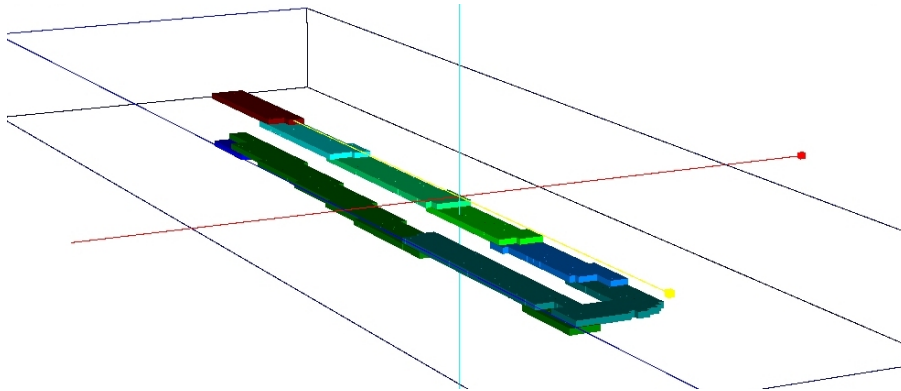
(b) 3-D Evidence Grid with a trace of the robot's recorded altitude moving down the ramp.

(c) $2\frac{1}{2}$-D map (with SICK readings).

Figure 13: Ramp in back of Building 1 at NRL.

to any of the "levels" around it. In addition, the robot will not be able to use the map to determine actual obstacles within each level (also described in Section 4.2). Even at this coarse resolution, the size of the map is $256 \times 64 \times 16$ (262,144 cells).

Figure 13(c) is a map of the same ramp created using the $2\frac{1}{2}$-D map structure. By obtaining the maximum height of an occupied space at each location, the robot can then use that information to determine where it can and cannot move. As described in Section 4.2, the height that the ATRV-Jr. can climb is about 0.1 meter. Therefore, if it calculates a curb to be less than or equal to 0.1 meter, then the robot will determine that it can move onto that curb. However, if the robot calculates the curb to be greater than 0.1 meter, then it will mark that location as an obstacle and will not try to move through that area. This method can also be used to determine negative obstacles such as cliffs. If a drop in elevation is beyond what the robot can handle, it will mark that area as a non-traversable boundary.

By calculating the difference in height between consecutive cells in the map representation, the robot can determine both positive and negative obstacles that it cannot move past or on. For the ramp, the elevation changes between the cells will be below the maximum climbing height of the robot (provided that the x and y resolutions are fine enough), allowing it move up and down a ramp without any problems. Figure 14 was created using this method. For each cell in Figure 13(c), the maximum distance (in the $-z$ direction) of that cell's height to the heights of all its neighbors is used to determine if an obstacle exists or not. Any neighbors with a height value greater than the current cell
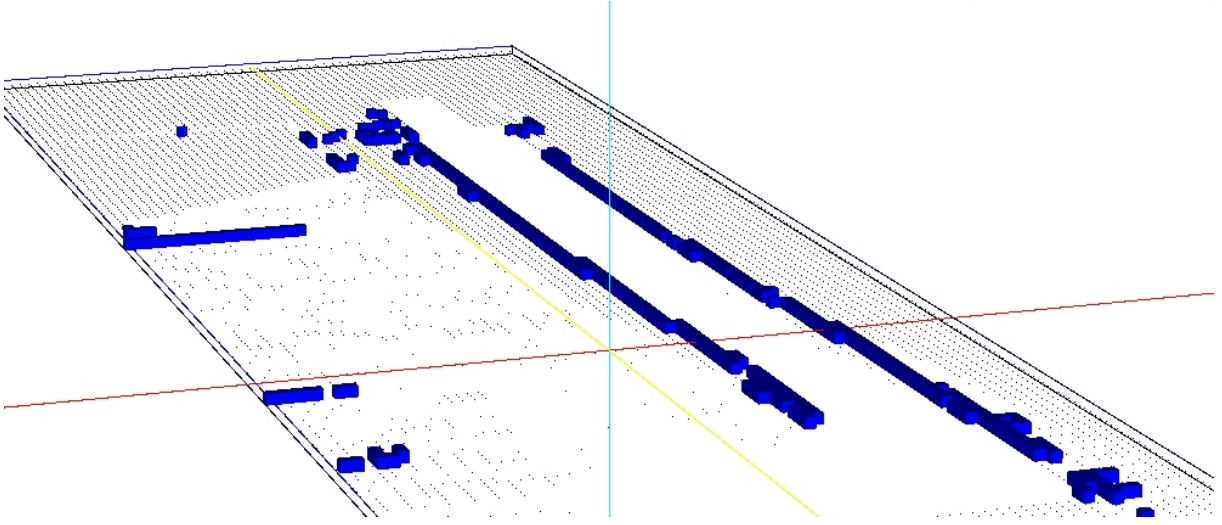
Figure 14: Converted $2\frac{1}{2}$-D map.

is ignored and the difference will be detected when that higher cell value is computed for the maximum $-z$ distance. If the resulting absolute distance is greater than the robot's maximum climb height, then the cell will be marked as an obstacle—otherwise, it will be marked as a traversable area. Even though it might only be the boundary between two cells that is the actual barrier from one location to the next (e.g. a cliff), and each cell could be used by a robot on either side of the barrier, the higher cell is still marked as an obstacle. Since each cell represents only a small portion of the world (0.125 meters in this map), it will not have a noticeable impact on limiting where the robot can move.

The maximum distance from the height of a cell to its neighbors is only determined in the $-z$ direction to prevent marking the obstacle boundary twice—and at the lower height (if there was a cliff, taking the $-z$ value would mark the boundary at the top of the cliff versus at the bottom). Figure 14 is similar to Figure 10 (the 2-D evidence grid) where the black dots represents areas where the robot has not "explored" yet, the

white areas represent unoccupied space, and the blocks represent occupied space. Using the information from the $2\frac{1}{2}$-D map structure (and the obstacle determination method just described), the robot is able to determine where obstacles and traversable areas are in a 3-D world. Finally, the size of this $2\frac{1}{2}$-D map is only $256 \times 64 \times 2$ (32,768 cells—as compared to the 262,144 cells needed by the 3-D evidence grid). By identifying areas where it can and cannot move to, the robot can now steer clear of any obstacle that it can detect (obstacle avoidance).

Some robots are able to descend larger distances than they can climb. However, for the ATRV-Jr., doing so would mean an uncontrolled drop. Taking into account the 2 SICKs (each weighing 4.5 kg) located in front, the robot would probably land on its "face" if it was unable to control its descent. For the robots that do have different maximum climbing and descending distances, the information in the $2\frac{1}{2}$-D map can be used to calculate obstacles based on the direction the robot is traveling. Since this does not apply to the ATRV-Jr., an actual method to account for such a scenario was not developed by this project.

4.3.2   Path Planning

Using the method discussed in Section 4.3.1 to determine areas where the robot can and cannot move to, the $2\frac{1}{2}$-D map structure can also be used to generate paths to a goal using the TRULLA algorithm described in Section 2.2.4. The converted $2\frac{1}{2}$-D map (Figure 14) provides TRULLA with all the information it needs to perform its path planning

algorithm. The areas labeled as traversable in Section 4.3.1's $2\frac{1}{2}$-D map will be treated the same as the areas labeled "empty spaces" in the 2-D evidence grid. The algorithm begins at the goal and propagates to neighboring traversable cells in the same way as in the 2-D case. As TRULLA "radiates" its field lines from the goal, all "unoccupied" space which can be reached from the goal (not completely blocked off by the "occupied" spaces) will be marked with a path to the goal. Whether or not the unknown spaces will serve as a blocked or traversable area can be decided by the user. For a robot moving in 3-D, it will be best if these areas are considered blocked (since those areas might hide a large enough drop to damage the robot). However, if no paths could be found when blocking the unknown areas off, then the robot will need to explore and identify these areas to see if it can find a path through there to the goal. Using this method (as illustrated in Figure 6), all traversable areas which are able to reach the goal will contain a path that is capable of directing a robot to the goal.

In addition, TRULLA also allows cells to be weighted (to slow the wavefront propagation of a path through certain areas so that a more desirable path would reach a cell first (as explained in Section 2.2.4)). In the converted $2\frac{1}{2}$-D map, the weight of each cell can be the height difference between the current cell and the next. For example, if the robot wanted to move one cell unit from its current location in the direction with the least effort (most desirable path), it would calculate the height difference between its current height and the height of its neighbors. If the robot is at height 3 with neighbors of heights of 2, 3, and 4 units, the difference (and weight for each direction) would be -1, 0 and 1.

Using the weights to represent the amount of effort needed to move to that area, the robot would in this case move to the cell with the height of 2 units. This weighing system which uses the information from the $2\frac{1}{2}$-D map structure can help TRULLA determine the more desirable path in a rugged environment (with multiple curbs, hills, ramps, etc.). The TRULLA algorithm demonstrates the ability of the $2\frac{1}{2}$-D map structure to provide a robot with the information needed for path planning.

4.3.3   Continuous Localization—The Room

To develop and test the Continuous Localization method used for this map structure, a long-term map of the environment needs to be created. A robot needs to be able to perform continuous localization and frontier exploration before it is able to generate its own map. Meanwhile, the development of its continuous localization algorithms requires the existence of a long-term map for testing. To address this issue, a long-term map will be created by hand. Since the purpose of continuous localization is for the robot to be able to update its location based on a reliable long-term map, the map that needs to be created will have to be as accurate as possible. In order to generate this developmental long-term map, the exact location of the robot must be known when incorporating scans into the map. Because it is much easier to determine the exact location (x, y, and z) of the robot in the laboratory (the floor tiles are all a foot in length and the altitude is relatively the same) as opposed to the outside ramp, the robot was moved back in the laboratory for the development of this part of the project. By continuously providing the
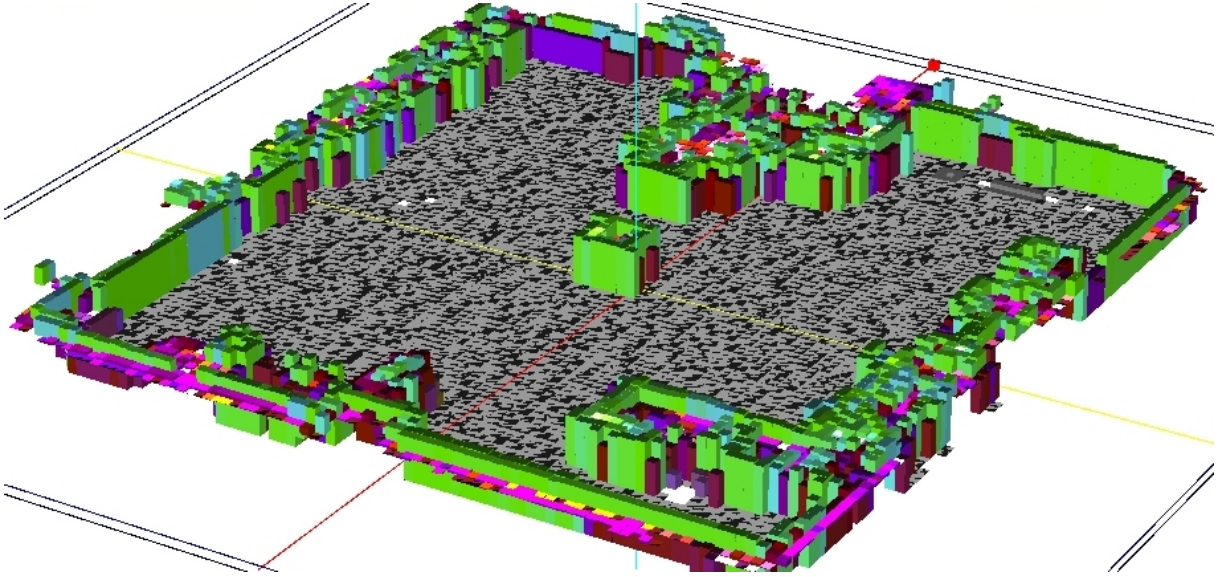
Figure 15: $2\frac{1}{2}$-D Long-Term Map of Room 105.

robot with its exact location in the room, it is able to develop an accurate long-term map from its sensor readings. Figure 11(a) is a panoramic view of Room 105 (in Building 1 at NRL—represented by Figures 10, 11(b), and 11(c)) and Figure 15 is the long-term $2\frac{1}{2}$-D map structure of that room. This map representation is created using a $128 \times 128 \times 2$ size grid—double the size of the 2-D map in Figure 10 but four times smaller than the 3-D map in Figures 11(b) and 11(c).

In continuous localization, the short-term maps are compared to the long-term map using a registration process (Figure 5) to determine the corrections needed to align the two maps together. The robot takes a short term map and tries to find a matching location on its long term map. It does this by searching an area in a fixed range around its current estimated location. This range varies between robots. The registration process will obtain a match result for a number of shifts along the x and y axes and rotation along the z axis

from the point where the search was begun. The match score for each map comparison is determined by taking the sum of the absolute difference between valid cell heights of the two maps. A low match score will represent a close match whereas a high match score will indicate large height differences between the two maps (a poor match). Once the results from all the combinations of shifts and rotations are obtained, the data is then inverted, normalized, and raised to the fourth power. By inverting the match scores, the correction with the highest match will now receive the the higher score—likewise, the correction with the lowest match score will receive the lowest value. Normalizing allows the range to be reduced to a more usable form (from 0 to 1), and raising it to the fourth power allows us to clearly identify the best matches and cut out the poor ones. If a few possible registrations were found to produce a good match score, then the weighted average of these (based on the actual score) is taken to determine the x, y, and yaw correction to update the robot's location. Using the x, y correction, along with the location that the robot thought it was at during the time it initiated the registration process, the actual x and y position of the robot can be determined. Going to that cell location in the long-term map will provide the actual z location of the robot at the time of registration. Taking the difference between the actual z value and the height at which the robot thought it was will yield the z correction for the robot. Since the values for *roll* and *pitch* are taken directly from the sensor, they are not subject to any systematic errors and are not corrected for in this registration. Errors that need be corrected by the registration process are the ones which accrue as a result of using values which are calculated based on its prior values. Since

roll and *pitch* are taken directly from the sensor and are only subjected to sensor noise (not previous errors), they do not need to be accounted for by this registration process since the errors will cancel each other out over time. All of the other corrections (x, y, z, and *yaw*) will be determined by the registration process and will be used to update the robot's location.

Figure 16 follows a remotely controlled robot, running continuous location, in the room represented by Figure 11(a) (using Figure 15 as its long-term map). Figure 16(a) displays the robot's starting location in the room (it is provided with its exact position in the room). In Figure 16(b), the robot performs its first localization. The darker robot shadow represents the estimated location of the robot without the localization correction. The lighter robot image represents the corrected position. The difference between the two robots is most visible in Figure 17(b). In the other figures, the difference between the two positions is small enough that the robot's slightly overlap. Figures 16(a)-17(b) follows the robot as it travels around the room—displaying the difference between the robot's corrected and uncorrected estimates of its position. Figure 17(b) shows the robot after one lap around the room. After comparing the robot's actual position in the room with its corrected and uncorrected estimates, it was determined that the estimated position of the robot (with correction) placed it 0.047 meters from its actual position (with a 1° rotation error) while the uncorrected estimate was off by 0.693 meters (with a 6° rotation error). From this test, it is determined that the robot is able to localize itself using a $2\frac{1}{2}$-D map structure within the tolerance range of the map (roughly the size of a cell—since
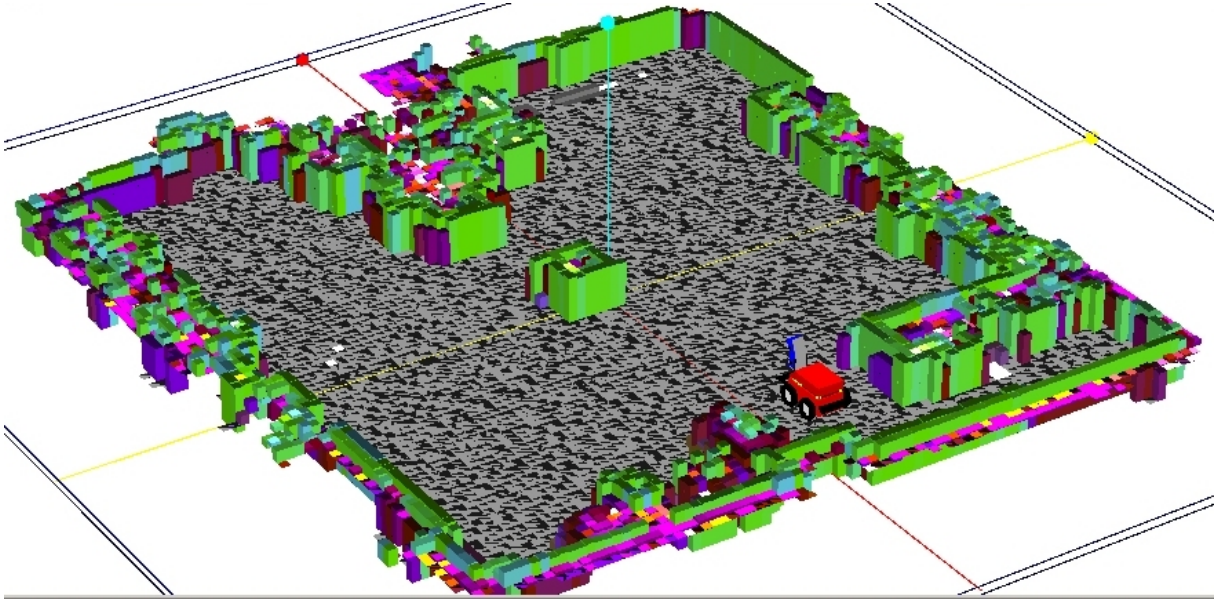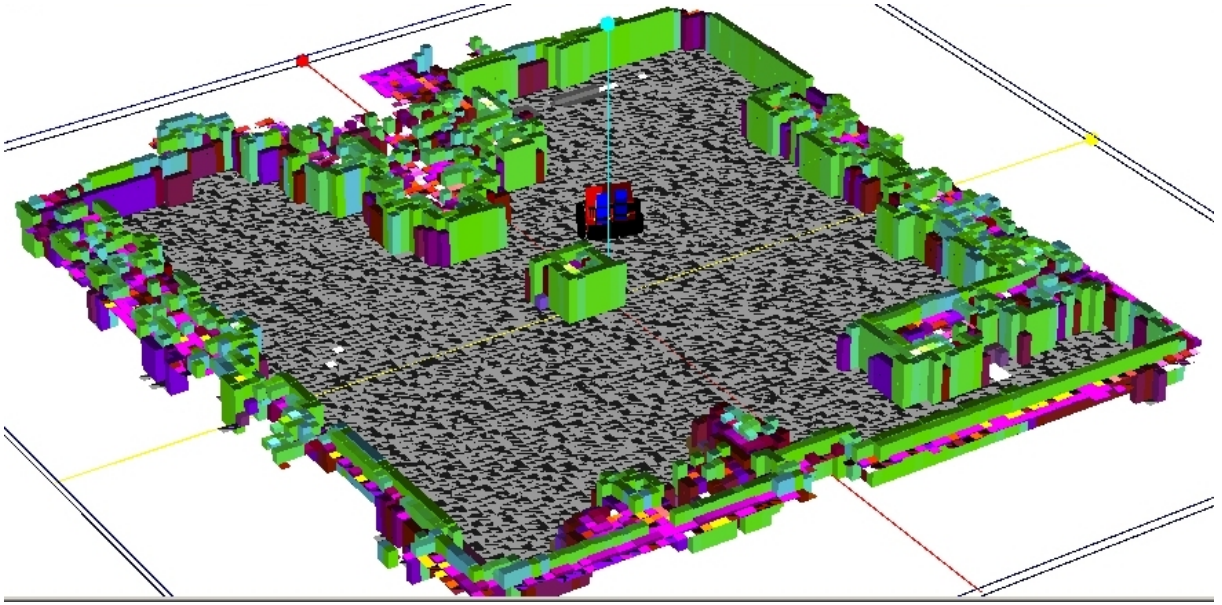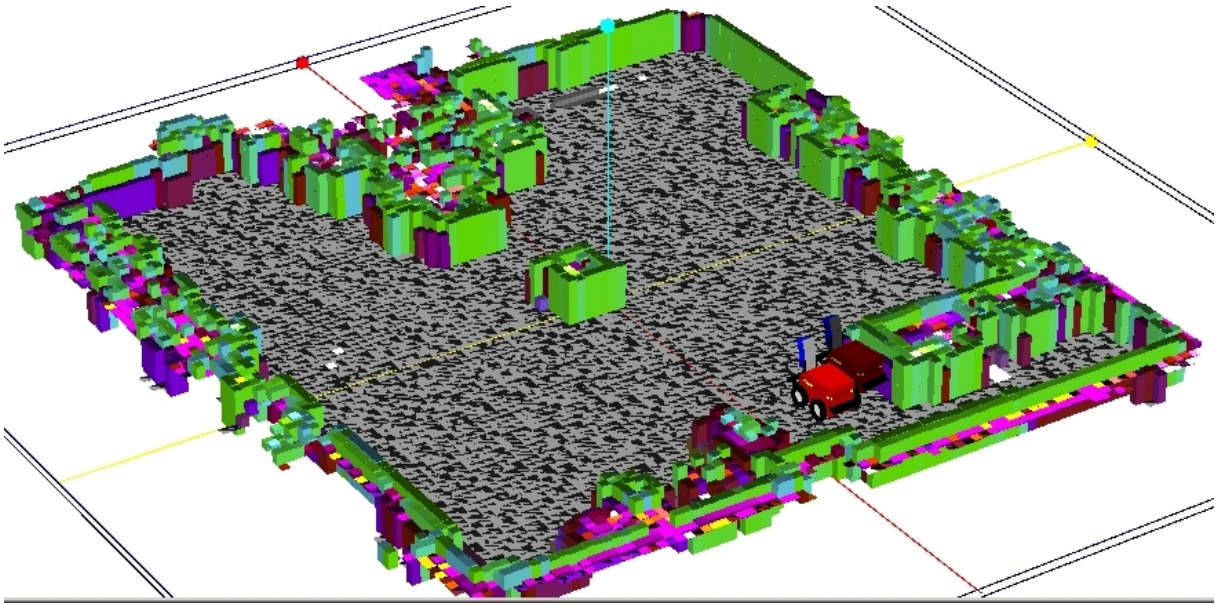
(a) Initial Starting position in Long-Term Map.
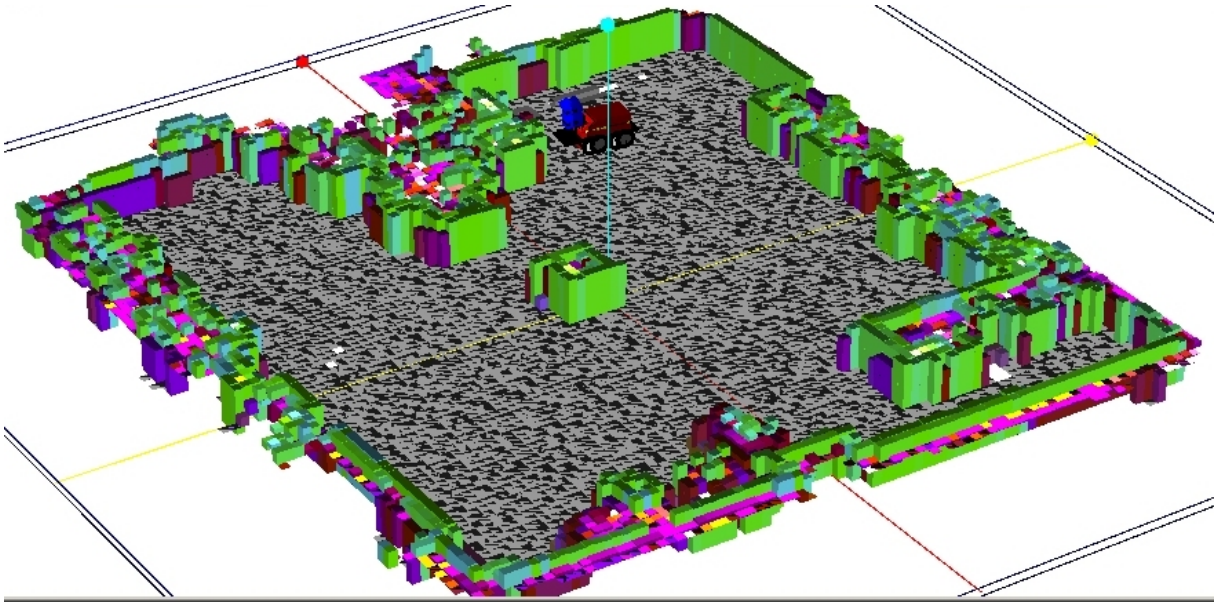
(b) After First Correction.

Figure 16: Continuous Localization using the $2\frac{1}{2}$-D Map—Part 1.

(a) After Second Correction.

(b) After 14 Corrections.

Figure 17: Continuous Localization using the $2\frac{1}{2}$-D Map—Part 2.

that is the accuracy of the map).

4.3.4   Limitations

Even though the $2\frac{1}{2}$-D map structure is shown to provide the robot with the ability to localize itself, the experiment was conducted in a laboratory room. Whether or not the the map's localization capabilities will work in the presence of varying ground elevation (such as the ramp) was not tested. Therefore, it cannot yet be said for certain that this map structure will truly localize a robot in a 3-D environment.

In addition, as the $2\frac{1}{2}$-D map structure currently stands, it is unable to represent multi-layered environments such as overpasses or bridges. By only representing one height measurement for each x-y position, the robot will not be capable of storing more than one path through a given area. This predicament prevents the robot from obtaining an accurate representation of its traversable environment (and might cause it to run into obstacles above it). Though this map structure will allow a robot to move in areas of varying elevation, it does not provide the robot with the ability to store more than one level of information at a certain x-y position. Such an issue must be resolved before a robot can create an accurate representation of its surroundings since overlapping structures do exist in the environment that this project was created for.

4.3.5   Exploration

Considering the limitations mentioned in Section 4.3.4, if it can be shown that a robot can use the $2\frac{1}{2}$-D map structure to localize itself while moving around from one ground

elevation to another, then it can be argued that this type of map structure will allow a robot to explore an outdoor environment which does not contain any overhead structures (e.g. bridges, archways, etc.). For a robot to perform the Frontier Exploration described in Section 2.2.3, it needs to determine where its frontiers are, plan paths to reach those frontiers, and localize itself along the way. The ability of a robot to perform the latter two tasks are presented in Section 2.2.4 and 4.3.3 respectively. Determination of the map's frontiers can be performed in the same way that is done for a 2-D evidence grid. The converted $2\frac{1}{2}$-D map displayed in Figure 14 can be interpreted to represent the same information as a 2-D evidence grid (occupied space, open space, and unknown space). The frontiers are easily identified as the boundaries between open and unknown space. As a result, if the conditions mentioned earlier are true, then the robot will be capable of performing Frontier Exploration.

5   Future Work

5.1   Testing in a 3-D Environment

There are two main issues which must be resolved before this $2\frac{1}{2}$-D map structure can be used to represent a 3-D environment for an autonomous robot. The first is to determine whether this map structure will allow a robot to localize itself while its movement causes its elevation to change. In order to do this, a map of an environment with varying altitudes must be created and given to the robot. For this project, an accurate map of such an

environment was unable to be created due to the requirements for such an undertaking. As described in Section 4.3.3, the creation of an accurate map requires the exact location and orientation of the robot to be known in order to update the right cells with the correct height from the sensor readings. For the laboratory room, this was a fairly easy task since the elevation remained the same (z was constant) and the floor tiles provided a method for easily determining the robot's x and y position. Modeling an outside environment proves to be more difficult. Not only must the x and y distance of the robot be determined, but the z coordinate must also be measured with regards to some location in space designated as the map origin. Since there is no easy (cheap and readily available) way to acquire these measurements, the ability for the robot to navigate in such an environment using the $2\frac{1}{2}$-D map can not be determined. An alternative to actually measuring the the x, y, and z location in an outside environment can be provided by building a representation of such an area (with ramps and other changes in elevation) in the lab. By doing so, the x, y, and z position of the robot could be determined more easily by using some of the characteristics of the room to help measure distances. These items can include the markings of the tiled floor and a constant level from which z can be determined (i.e. floor or ceiling).

5.2   Multi-Layered Maps

The next issue that must be explored is the ability of the robot to represent a multi-layered environment—an environment in which the robot can be positioned at the same

x and y coordinates but at different altitudes (such as being on or under a bridge). As the current $2\frac{1}{2}$-D map structure stands right now, it is not capable of performing this task. A full 3-D evidence grid should be able to perform this task since it represents every cell in space. However, for the reasons explained in Section 4.2, such an attempt would be infeasible. One solution might be to stack multiple $2\frac{1}{2}$-D maps on top of each other—having each one represent an altitude range equal to the height of the robot. The taller the robot, the more altitude each map will cover. Since the robot cannot move to two different z values at the same (x,y) location that are separated by a distance smaller than the height of the robot (or else the robot at the lower z level will be overlapped by the object on which the robot will be on if it was at the higher location), each individual $2\frac{1}{2}$-D map will only need to represent the one layer that the robot can possibly be on for that altitude range. The $2\frac{1}{2}$-D map has been shown to have the capability of representing the elevation of an area with a potential accuracy of 1 millimeter. By stacking these $2\frac{1}{2}$-D maps, a structure similar to a 3-D evidence grid can be created where every z-layer is now taller in size (requiring less space), more accurate in representing altitude variations between cells (allowing for better detection of obstacles), and the map will have the ability to represent multi-layered environments.

5.3   Improving Accuracy and Reliability

Other areas that can be researched and made more accurate include augmenting the continuous localization algorithm to make it more robust, creating a better sensor model,

and fine-tuning the method used to convert the $2\frac{1}{2}$-D map into a form similar to the 2-D evidence grid. As mentioned in Section 2.2.3, the continuous localization algorithms are not too robust. If the registration process returns an inaccurate correction for the robot's position, subsequent corrections should be able to account for that error. If it does not, then the errors will eventually compound and grow larger until the robot is completely lost. A more robust continuous localization algorithm should be able to detect a bad correction and perform a more thorough search.

In this project, the values for the sensor model were roughly determined based on a few trials and observations. Though the model used did produce fairly decent results, developing a more accurate model can greatly improve the accuracy of the sensor data and therefore the map.

One way to more accurately use the data from the $2\frac{1}{2}$-D map would be to change the way it is converted into the structure similar to the 2-D evidence grid. The current algorithm implemented for the conversion uses the maximum climbing height of the robot to determine the cutoff point when assigning traversable space versus a blocked boundary. Any difference in height between a cell and its neighbor that is greater than the cutoff value is assigned an occupancy probability of 1 (indicating a obstacle). Any value below or equal to the cutoff value is given an occupancy of 0 (traversable area). Taking into account the sensor limitations and noise, it might be better to use the height difference to determine a probability of occupancy instead of just assigning 0's and 1's. This would prevent a height difference value, which might be just a small amount over the cutoff

(due to sensor noise), from being immediately labeled as an obstacle. By calculating an actual probability based on the height difference, the converted map can be more robust in handling sensor noise and some errors.

6 Conclusions

The $2\frac{1}{2}$-D map structure that was developed for this project was shown to reduce the size required to store 3-D information of an environment with varying ground elevations. This was done by recording the altitude of the highest object at each location in the world (whether that would be the the ground or some obstacle at that (x,y) location). Using this map structure, the robot was shown to be capable of generating a map of an outdoor ramp and a laboratory room. Though the map of the ramp was created through a rough scan, the robot was still able to use that map to determine areas where it can and cannot move around in. This provided the robot with the ability to determine where the obstacles (anything it can't climb over or roll down) are in a 3-D world. By being able to provide information regarding the location of traversable and blocked areas, the $2\frac{1}{2}$-D map structure is capable of being used when implementing path planning algorithms (such as TRULLA). An experiment was also performed which showed that the map structure was also able to be used by a robot for running continuous localization. Even though the robot's ability to localize itself using the $2\frac{1}{2}$-D map was not fully demonstrated, it was able to do so in the lab.

It appears that this map structure is able to provide a robot with the ability to

move autonomously in a 3-D environment since it satisfies the four required abilities of a mobile robot—exploration and map making, localization, path planning, and obstacle avoidance. However, there are two issues which must be addressed concerning this $2\frac{1}{2}$-D map structure. The first is the need for an actual 3-D test of the robot's localization abilities. To perform this experiment, an accurate map of the 3-D environment needs to be created. The ability to do so was beyond the resources available for this project and therefore was not performed. The other issue deals with mapping in a multi-layered environment. As the map structure currently stands, it is not capable of representing more than one layer of travel for the robot. A consideration for future work would be to stack multiple $2\frac{1}{2}$-D map structures together to represent multiple overlapping layers for a robot. Since the height of each individual $2\frac{1}{2}$-D grid in the stack will be equal to the height of the robot, more space is represented in a fewer number of cells (as compared to a 3-D evidence grid) while providing the accuracy of the $2\frac{1}{2}$-D grid.

The $2\frac{1}{2}$-D map structure is capable of performing the four basic abilities required of a mobile robot. It still needs further testing of localization in a full 3-D environment, however the results shown here indicate that it should perform suitably in that task. The model also needs to be extended so that it can represent multiple overlapping layers. The extension would complete the system, and the robot could then use it to navigate itself in a 3-D environment.

References

[1] Arkin, Ronald C. <u>Behavior-Based Robotics</u>. Cambridge: MIT Press, 1998. (pp. 18-19)

[2] Borenstein, J., an Koren, Y., "Histogramic In-Motion Mapping for Mobile Robot Obstacle Avoidance," *IEEE Transactions Robotics and Automation*, vol. 7, no. 4, August 1991. (pp. 535-539)

[3] Graves, Kevin. <u>Continuous Localization and Navigation of Mobile Robots</u>. Annapolis: US Naval Academy Trident Report, 1997. (pp. 8, 12)

[4] iRobot™. <u>The ATRV-Jr™ Mobile Robot</u>. 30 Nov. 2001

<http://www.irobot.com/rwi/p03.asp>

[5] Liu, Y., et al. (2001) "Using EM to Learn 3D Models of Indoor Environments with Mobile Robots." *Eighteenth International Conference on Machine Learning*, Williams College.

[6] McKerrow, P.J. <u>Introduction to Robotics.</u> Boston: Addison Wesley, 1991.

[7] Moravec, H. & Elfes, A. (1985). High resolution maps from wide angle sonar. In *Proceedings of the IEEE International Conference on Robotics and Automation*, (pp. 2833-2839), Leuven, Belgium,.

[8] Moravec, H.P., "Sensor Fusion in Certainty Grids for Mobile Robots," *AI Magazine*, vol. 9, no. 2, Summer, 1988. (pp. 61-74)

[9] Murphy, Robin. <u>Introduction to AI Robotics</u>. Cambridge: MIT Press, 2000. (pp. 16,32, 41, 236, 367, 380, 386)

[10] Murphy, R., et al. (1999) "Integrating Explicit Path Planning with Reactive Control for Mobile Robots using Trulla." In *Robotics and Autonomous Systems*. (pp. 225-245)

[11] Shafer, G., <u>A Mathematical Theory of Evidence</u>, Princeton University Press, 1976.

[12] Tesler, Pearl. <u>Universal Robots: The History and Workings of Robotics</u>. 2000. 01 Apr. 2002 <http://www.thetech.org/robotics/universal/index.html>

[13] Wampler, C., "Teleoperators, Supervisory Control," *Concise International Encyclopedia of Robotics: Applications and Automation*, R. C. Dorf, editor-in-chief, John Wiley and Sons, Inc., 1990. (pp. 997)

[14] Yamauchi, B., et al. (1999) "Integrating Exploration and Localization for Mobile Robots." In Adaptive Behavior. (pp. 217-230)